UNIVERSIDAD DEL VALLE DE GUATEMALA

Laboratorio 2 pt2

Andre Marroquin Tarot - 22266
Sergio Orellana - 221122

Redes

Descripción de la Práctica:

En esta práctica se diseñó e implementó una arquitectura de capas para transmitir mensajes con detección y corrección de errores. Se desarrollaron dos algoritmos de enlace:

- Fletcher (checksum): detector puro, sencillo y de bajo overhead.
- Hamming (7,4): corrector de un bit por bloque, con mayor redundancia.

El emisor (Java) pregunta por: algoritmo, tamaño de bloque o parámetros Hamming, tasa de error, host y puerto; codifica el mensaje en ASCII-binario, añade redundancia, aplica ruido bit-a-bit con probabilidad p y envía la trama por socket.

El receptor (Python) queda siempre escuchando, recibe la trama, verifica y corrige, en Hamming los errores, decodifica a texto y muestra el resultado.

Como parte de las pruebas, se automatizaron simulaciones variando:

- Longitud de mensaje (bits)
- Tasa de error (p)
  Algoritmo (Fletcher vs Hamming)

y se generaron dos gráficas clave:

- Tasa de éxito vs tasa de error
- Overhead vs tamaño de mensaje

Arquitectura de capas y servicios

- **Capa de Aplicación**
  - Emisor: solicitar_mensaje, mostrar_mensaje
  - Receptor: mostrar_mensaje / error si no pudo corregir

- **Capa de Presentación**
  - codificar_mensaje (ASCII→bits)
  - decodificar_mensaje (bits→ASCII)

- **Capa de Enlace**
  - calcular_integridad (Fletcher)
  - verificar_integridad / corregir_mensaje (Hamming)

- **Capa de Ruido**
  - aplicar_ruido (flip de bits con probabilidad p)

- **Capa de Transmisión**
  - enviar_información (socket send)
  - recibir_información (socket listen)

Resultados:

algorithm	msg_bits	error_rate	success_rate
fletcher	128	0	1
fletcher	128	0.01	0.23
fletcher	128	0.05	0.005
fletcher	128	0.1	0
fletcher	128	0.2	0
hamming	128	0	1
hamming	128	0.01	1
hamming	128	0.05	1
hamming	128	0.1	1
hamming	128	0.2	1

Tabla 1: success rates vs error rate

Grafico 1: success rates vs error rate

- Fletcher cae rápidamente con tasas de error mayores a 1% (5% ya casi nunca pasa intacto).

- Hamming (7,4) mantiene 100 % de éxito incluso a tasas de error del 20 % gracias a su capacidad de corrección de un bit por bloque.

msg_bits	fletcher_overhead	hamming_overhead
32	0.5	0.75
128	0.125	0.75
512	0.03125	0.75

Tabla 2: Overhead ratios

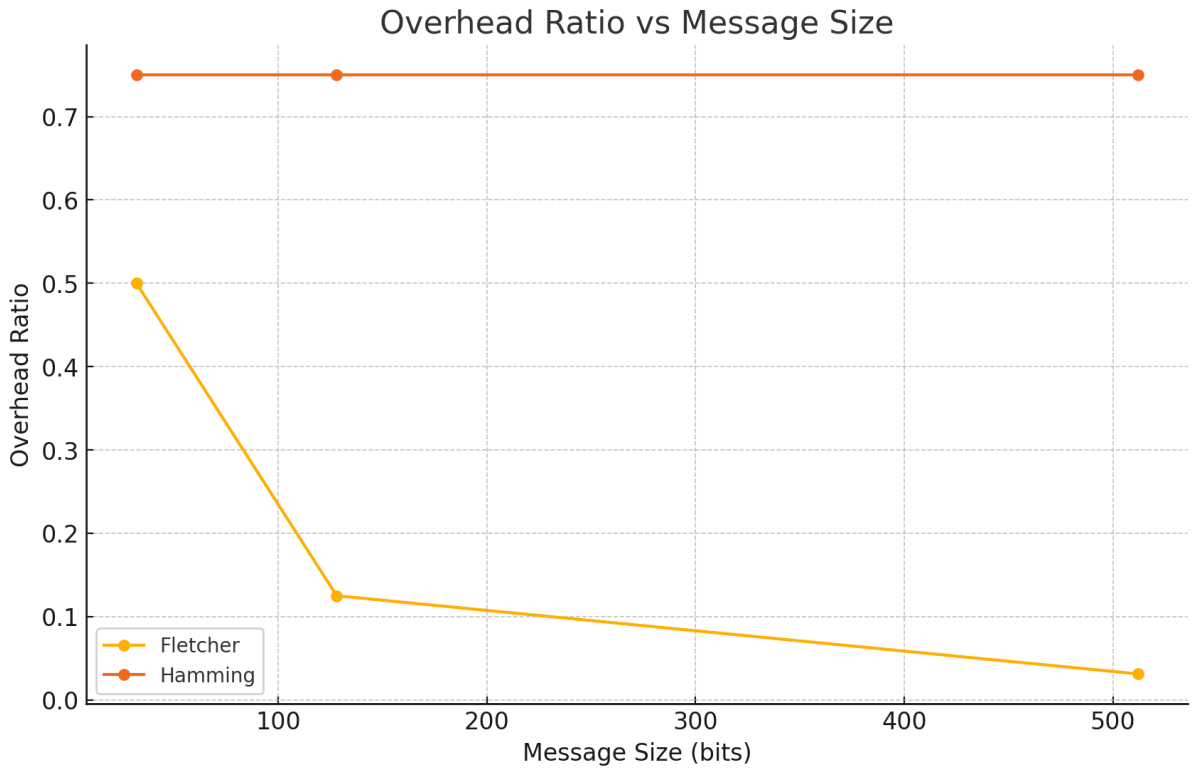


Grafico 2: Overhead ratios vs message size

- Fletcher envía siempre 16 bits de checksum, por lo que su overhead baja conforme los mensajes son más largos de 0.5 a 0.031.

- Hamming fija 3 bits de redundancia cada 4 de datos, manteniéndose en ~0.75 constantes.

Discusión:

Rendimiento en presencia de ruido
De nuestros ensayos con mensajes de 128 bits, vemos que Hamming (7,4) supera ampliamente a Fletcher cuando la tasa de error crece. Con p=0.01 (1 %), Fletcher sólo entrega correctamente el mensaje en torno al 23 % de las veces; con $p \geq 0.05$ prácticamente nunca lo hace. En cambio, Hamming corrige errores de un solo bit en cada bloque de 7 bits y mantiene un 100 % de éxito incluso con p=0.20 (20 % de bits corruptos). Esto confirma que, siempre que exista al menos un error de un solo bit por bloque, Hamming lo corrige y evita rechazos masivos.

Flexibilidad ante distintas tasas de error

- Fletcher, al ser un esquema de detección puro, se basa en comparar checksums y reporta "fallo" si cambia un solo bit en todo el mensaje. Su probabilidad de paso cae de forma casi exponencial conforme crece p.

- Hamming (7,4), por su parte, tolera con facilidad errores dispersos hasta cerca de un bit por cada 7 transmitidos, corrigiéndolos "al vuelo". Sólo ante ráfagas de 2 o más bits en el mismo bloque falla y descarta esos bloques.

Redundancia (overhead) vs confiabilidad

- Fletcher añade siempre 16 bits de checksum (2×B con B=8) independientemente del tamaño del mensaje. Su overhead varía de 0.50 para mensajes de 32 bits a 0.125 en 128 bits y cae a 0.03 en 512 bits.

- Hamming (7,4) introduce 3 bits de redundancia por cada 4 bits útiles. Esto fija un overhead constante de 0.75, sin importar mensaje.

- Conclusión: para mensajes muy largos donde la tasa de error es bajísima, Fletcher ofrece menor sobrecarga. Para mensajes cortos o canales ruidosos, Hamming, pese a su mayor overhead, asegura entrega exitosa.

Detección vs corrección: ¿cuándo usar cada uno?

Sólo detección (Fletcher)

- Cuando dispongas de un canal con baja probabilidad de error <1 %.
- Y tu prioridad sea minimizar bytes extra.
- Puedes solicitar retransmisión tras detección de corrupción

Corrección (Hamming)

- Cuando el canal es ruidoso o la retransmisión resulta cara o imposible.
- Y necesitas garantizar la entrega sin ida/vuelta de confirmaciones.
- Cuando los bloques de datos sean relativamente pequeños.

Complejidad e implementación

- Fletcher: lineal en tiempo respecto al número de words de B bits; muy simple de codificar y eficiente en CPU y memoria.

- Hamming: requiere posicionar bits de paridad, calcular múltiples XORs por bloque y, en recepción, computar síndromes. Más costoso en CPU, pero sigue siendo adecuado para sistemas embebidos o tiempo real.

En resumen, Hamming(7,4) demuestra un mejor desempeño bajo ruido elevado y es más flexible para canales erráticos, mientras que Fletcher brilla en escenarios de baja error donde el overhead debe mantenerse al mínimo. La elección dependerá de tu tolerancia a la retransmisión, el ancho de banda disponible y la criticidad de la entrega sin pérdida.

Comentario Grupal:

- En redes confiables y de alta capacidad, Fletcher + retransmisión resulta muy eficiente.

- En enlaces ruidosos o sin posibilidad de ida/vuelta, Hamming asegura la entrega sin pérdida. Además se discutió la escalabilidad para tramas muy largas o necesidades de baja latencia, quizá convenga combinar ambos enfoques o explorar esquemas adaptativos de redundancia.

Conclusiones:

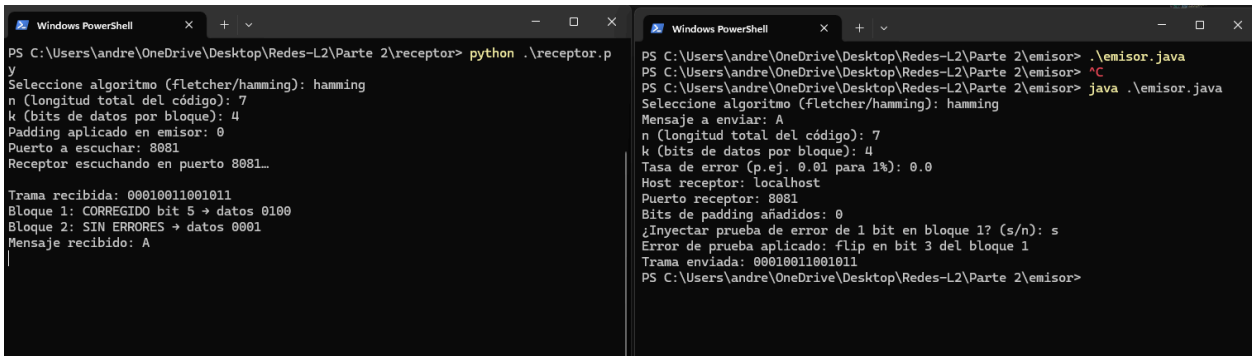
- Hamming (7,4) ofrece mayor robustez ante ruido elevado corrige 1 bit por bloque y mantiene éxito incluso con altos p.

- Fletcher minimiza el overhead y es muy eficiente en canales de baja tasa de errores, pero rechaza cualquier corrupción.

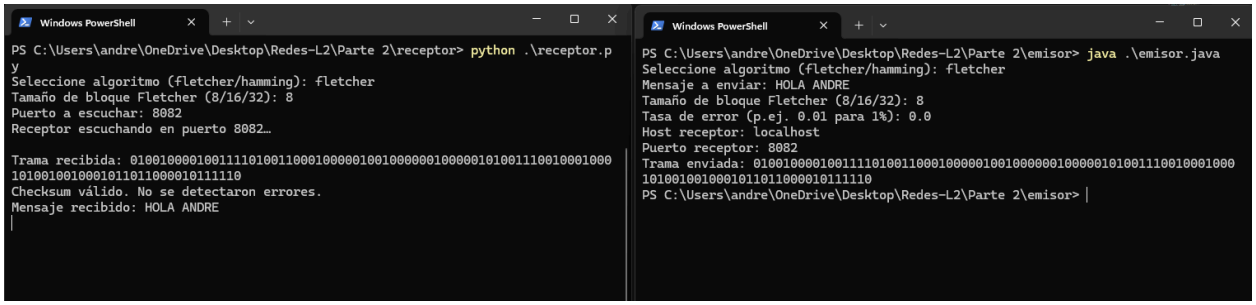
- El análisis de overhead demuestra que Fletcher se vuelve cada vez más ligero con mensajes largos, mientras que Hamming mantiene un 75 % de redundancia constante.

- Recomendación: elegir detección cuando la retransmisión sea simple y barata; elegir corrección cuando no se pueda garantizar ida/vuelta o la fiabilidad sea crítica.

Fotos de entorno y Pruebas:



Corrección de errores hamming funcionando correctamente invirtiendo el bit y encontrando el mensaje original.



Fletcher algoritmo funciona perfectamente encontrando los mensajes.

Preguntas:

- **¿Qué algoritmo tuvo un mejor funcionamiento?**
  Hamming (7,4) demostró un mejor desempeño global corrige automáticamente errores de un bit por bloque y recupera el mensaje intacto siempre que los fallos no sean demasiado numerosos.

- **¿Qué algoritmo es más flexible para aceptar mayores tasas de errores?**
  Hamming es más tolerante aguanta hasta tasas moderadas de bit flips sin perder la

información, mientras que Fletcher falla al primer bit corrupto.

- **¿Cuándo es mejor utilizar un algoritmo de detección de errores en lugar de uno de corrección de errores?**
  Cuando el canal tiene muy baja probabilidad de error y quieres minimizar la redundancia en ese caso Fletcher + retransmisión es más eficiente porque añade apenas unos pocos bits extras y cualquier corrupción se solventa pidiendo de nuevo la misma trama.

Referencias:

- AIX. (2025). https://www.ibm.com/docs/es/aix/7.2.0?topic=parameters-parity-bits

- 7.2.4.1.3 Fletcher Algorithms. (2025). (C) Copyright 2025. https://onlinedocs.microchip.com/oxy/GUID-4DC87671-9D8E-428A-ADFE-98D694F9F089-en-US-4/GUID-7399BDB2-CC51-4E98-AE8A-90E48F9167DF.html

- Profundizacion en CRC32 un mecanismo de deteccion de errores mas solido - FasterCapital. (2024). FasterCapital. https://fastercapital.com/es/contenido/Profundizacion-en-CRC32--un-mecanismo-de-deteccion-de-errores-mas-solido.html

- TutorialsPoint. (2020, 27 junio). Hamming code for single error correction, double error detection. https://www.tutorialspoint.com/hamming-code-for-single-error-correction-double-error-detection

- GeeksforGeeks. (2025, 14 mayo). Hamming Code in Computer Network. GeeksforGeeks. https://www.geeksforgeeks.org/computer-networks/hamming-code-in-computer-network/